UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/676,373	09/30/2003	Stefan Jesse	09700.0216-00	3224

60668 7590 07/10/2008

SAP / FINNEGAN, HENDERSON LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
07/10/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/676,373	JESSE ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12 June 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *6/12/08*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 6/12/08.

As indicated in Applicant's response, claims 10, 18 have been amended. Claims 1-22 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 8, 18 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 4, 12, 19 of copending Application No. 10,676,374 (hereinafter '374).

Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following example of conflicting claims.

As per instant claim 8, copending '374 claim 4 recites a first data model being used to derive an API and employing the API to access development objects. But '374 claim 4 does not explicitly recite (i) first model defining development objects as building blocks for the application; (ii) generate intermediate objects therefrom and (iii) using the set of intermediate objects as inputs to generate the API, the model including (iv) a first model customized extension used to implement a feature of the API such as an indication of a file border and (v) API enforces relationships and constraints defined in the first model.

However, '374 claim 4 recites a variation of the language in claim 8 for limitations (i) and (ii) via the recital of 'defining file borders comprising identifying of development objects to be included in a file … in the data model … to be children of the main …object that are not identified as main…objects', the intermediate objects being added objects to the file of the main object including development objects defined in the data model. As for the *constraints enforcing* limitation of (v) based (iii)-- *using the set of intermediate objects as input for the API generating*—'374 claim 4 includes file storing user-defined code associated with the main development object; and in view '374 teaching of a definition file (see '374 claim 4) in light of objects being defined -- in terms of parent/child relationship in '374 claim 3 – one of ordinary skill would recognize these defined objects as well-known interrelated model components viewed in a GUI development interface of '374, i.e. API - to necessarily support user's development via instantiating one such development GUI API for accessing model objects. And this as a whole would be equivalent to (iii) for enforcing constraints as suggested above, or otherwise obvious variation thereof. As for the *customizable extension* comprising a file border indication referred to as (iv), this is suggested in '374 reciting of 'defining file borders for

development', and storing development objects in a repository based on the file borders, and accessing these objects via the API (*); so that one skill in the art would be motivated to provide an extension structure obtained from the repository (e.g. template builder) in the course of the API derivation with utilizing of information in the '374 stored file-based repository for the derivation. That is, the information thus extended (e.g. via a template builder) from the stored model/repository regarding a particular file border identity would be used to support the creation of API parameter or attributes which would be needed to access the very components stored from the '374 defining of file borders, as purported by the endeavor described as (*) from the above.

As per instant claim 8, '374 claim 19 also recites API derived from a data model, file borders defined in the model, and user interface using the API to access development objects being stored in a repository. Claim 19 does not recite 'first language model with extension to implement API and file border; but based on '374 reciting of association between component and model class 'that associates a user interface to a ... application model', the extension by use of border file suggest the *extension* limitation of instant claim 8 to provide deriving of association between stored model objects; that is, obviousness in terms of instant claim 8 limitations such as API for 'enforcing constraints' and 'language extension' does apply here in view of the rationale set forth above.

As per instant claim 18, '374 claim 12 also recites an obvious language variant thereof in expressing 'receiving of a model' in a development method, a *language extension* for defining a model representing blocks in terms of 'component class' and 'model class' as well as their inter-association for developing an application (Note: this would be an obvious variation of

building block relationships among objects and, hence suggesting constraints thereof), deriving a API based thereon, and use the API for *enforcing constraints* in the model within the development of the application.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-9, 18-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Charisius et al., USPN: 2002/0108101, (hereinafter Charisius)

As per claim 1, Charisius discloses a computer program product, tangibly embodied in an information carrier, for developing an application, the computer program product being operable to cause data processing apparatus to:

receive a first model in a first language (e.g. model 204 – Fig. 2; Unified Modeling Language - para 0088-0089, pg. 13; Fig. 3, 5 - Note: graphical views of language representation , e.g. UML, being instantiated as from core model 302 **reads on** model being in first language whose Java classes are building blocks – graphical view 204 in Fig. 2, or package classes view in Fig. 5 – representing a given development instantiated to form the SCT model), the first model defining development objects representing building blocks for developing the application (e.g.

Fig. 12-18), relationships among developing objects, and constraints for developing the application (Note: standardized UML notations and language rules reads on constraints between UML objects);

generate a set of intermediate objects using the first model (e.g. templates, data structure – para 0089, pg. 13; symbols – Fig. 10A-B; data structure 2114 – Fig. 21B); and

using the set of intermediate objects, generate an API (e.g. Meta Model 200, interface 610, Fig. 6-7 – Note: using the packages from derived template and class symbols represented in UML model, along with graphical view of code objects to generate an metamodel within interface 610, including instance of RWI, IDE or SCI **reads on** API being instantiated using intermediate objects, i.e. an instance of TMM within Interface 610 for further tasks; para 0109, pg. 15 – Note: using a modifiable and dynamic instance of the XML structure diagram to build correspondence between a DTD model and target XML structure reads on input for a parser component of above API – see Fig. 7 – the parser instance or API needed for parsing intermediate objects or derived markup language -- see Fig. 26B-C; parser– see Fig. 24; step 2116, Fig. 21B) wherein the API enables accessing the development objects (e.g. Fig. 7; DE: *extract information from the model* – para 0065; *access information* – para 0068, pg. 4; para 0089-0090, pg. 13).

Charisius does not explicitly disclose that the instance of API generated from the first model enforces the relationships and constraints defined in the first model. But based on Charisius teaches additional modules to support the created API (see modules 704 -Fig. 7) and using a graphical representation to validate new entries in the temporary model, the concept that using the API instance (TMM associated with framework instance 610 and support modules) for

accessing more development objects (see Fig. 10A-B; para 0091, pg. 13) and for validating whether these (classes) objects are compliant to their being structured or logical sequence dictated from their source hierarchical relationship (see Fig. 13-19; messages … collaboration … desired operation … emphasize time ordering – para 0092-0094, pg. 13) is strongly suggested. It would have been obvious for one skill in the art at the time the invention was made to implement the Quality Assurance modules (para 0068, pg. 4) and the IDE support modules (Fig. 7) along with the UML-derived graphical view instance of the TMM (see Figure 3-6) to enforce relationships and constraints defined in the 'first model' as set forth above, because of the very nature of object relationship defined and regulated as UML and class inheritance due to the object-oriented nature of source or templatized classes as endeavored by Charisius use of IDE via support of the above auditing capabilities when generating OO objects in a editable and updatable viewer to achieve a correct state of a TMM (see Fig. 10 and 11) viewed within the interface 610 as purported by Charisius development cycle (see Figure 2 and Figure 7)

As per claim 2, Charisius discloses instructions to convert the first UML model to a second data model as a transient TMM and less specific proprietary form (e.g. Fig. 2; para 0087, pg. 12), wherein the set of intermediate objects is generated using this second data model (e.g. Fig. 10A-B - Note: dynamic TMM view incorporating additional objects, structure and OO symbols – see para 0059-0060, pg. 4; para 0088-0094, pg. 12-13 -- **reads on** using second model to generate more intermediate objects being a second and different form of language than first UML notation).

As per claim 3, Charisius does not explicitly disclose that the second language model comprises XML. Charisius discloses UML objects as identified objects to include in a XML

model (see Fig. 25 and related text) and further teaches a first DTD model (a first model depicting relationship among objects) being parsed to generate intermediate objects (e.g. Fig. 21-23 – Note: parsing of model DTD – para 0101, pg. 14 - into 'XML structure diagram' – para 0109, pg. 15 – reads on intermediate object to derive DTD model into XML schema structure in a different language) and derive a dynamically incremented second XML model. It would have been obvious for one skill in the art at the time the invention was made to implement the interface 610 (see Fig. 3, 7) by Charisius so that, when the first model is in a DTD-based language or a UML object construct as mentioned above, the instantiated API 610 includes a GUI viewer instance operating as a dynamic application interface as to enable the developer to dynamically validate entries into the "XML structure diagram" as intermediate object as mentioned above; that is, using the TMM/610 API instance as means to further validate the corresponding relationship or mappings between the first source DTD model, the derived mappings or tagged constructs – see para 0110-0117, pg. 15-17—and the target XML language as intended in Charisius' framework API using its source auditing capabilities (see Fig. 21-24; see Fig. 7)

As per claim 4, Charisius disclose wherein the first language is UML (e.g. para 0088-0089, pg. 13).

As per claim 5, Charisius discloses wherein the set of intermediate objects comprises Java objects (e.g. Fig. 5; Fig. 12-16; *java ...template* - para 0089-0090, pg. 13).

As per claims 6-8, Charisius discloses wherein the first language comprises a customizable extension (e.g. view 204, TMM 200, code editor 208 -Fig. 2; para 0064-0067, pg. 4); wherein the customizable extension is used to implement an additional feature of the API (e.g.

para 0064-0067, pg. 4; Fig. 9), wherein the additional feature comprises an indication of a file border (e.g. *file is new... file ... been updated* -- para 0090, pg. 13– Note: repository of model – Fig. 2-5 --having files being enlisted for a project and identified for its update status **reads on** model extension with indication to file borders including management or versioning metric).

As per claim 9, Charisius discloses wherein the API comprises a copy and paste operation (e.g. Fig. 12-18, 22, 23; para 0090-0094, pg. 13 – Note: customization via user interface (see GUI pane with toolbar) to create instance of API from the core API of Fig. 7 whereby the IDE enables modeling and delete/add of development objects reads on GUI API in which *copy and paste* are features operable user's modifications of a UML view).

As per claim 18, Charisius discloses a computer program product, tangibly embodied in an information carrier, for developing an application, the computer program product being operable to cause data processing apparatus to:

receive a data model defining development objects … relationships among … objects, and constraints for developing application (refer to claim 1) ;

derive an API based on the data model (refer to claim 1); and

use the API to perform operations on the development objects (e.g. refer to claim 2; *extract information from the model* – para 0065; *access information* – para 0068, pg. 4; para 0089-0090, pg. 13 – Notes: refer to claim 1).

But Charisius does not explicitly disclose that the instance of API generated from the data model enforces the relationships and constraints defined in the data model. The API enforcing limitation has been addressed as obvious in claim 1.

As per claim 19, Charisius discloses wherein the API comprises an interface layer (e.g. para 0064-0067, pg. 4 Note: RWI API reads on interface layer wherein diagrams can be user driven), a proxy layer (e.g. IDE API reads on proxy layer wherein information are channeled, extracted and filtered for the interface layer to used) a state layer (e.g. SCI API reads on state layer wherein data received as-is is just displayed for plain view and editing by the RWI).

As per claim 20, Charisius discloses wherein the operations comprise creating a new development object as a transient object (e.g. Fig. 12-18); and modifying the transient object until the transient object is committed to a persistent file (e.g. Fig. 12-18; ; para 0090-0094, pg. 13).

As per claims 21-22, Charisius discloses comprising instructions to destroy the transient object if a delete command is requested before the transient object is committed to a persistent file; and to mark the persistent file as deleted if a delete is requested after the transient object is committed to a persistent file (e.g. Fig. 2; Fig. 7; Fig. 10AB; Fig 24-26; para 0118-0119, pg. 17 – Note: use of TMM transient structure to enable modifying/removing – as in not marked for persisting or committed for file repository – using the created API when parsing DTD or XML data which are previously stored **reads on** modifying a transient object and generate code when such object is committed; while retrieving corresponding DTD/XML reads on reusable objects being committed for DB persistence from a previous development instance).

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 10-17 are rejected under 35 U.S.C. 102(b) as being anticipated by Charisius et al., USPN: 2002/0108101, (hereinafter Charisius).

As per claim 10, Charisius discloses a computer program product, tangibly embodied in an information carrier, for developing an application, the computer program product being operable to cause data processing apparatus to:

receive a first model in a first language … relationships among the development objects (para 0101, pg. 14);

generate (a set of intermediate objects …) using the first model (para 0109, pg. 15; Fig. 21B; Fig. 26B-C);

and generate an XML schema (e.g. Fig. 21-24) using the set of intermediate objects as inputs such that the schema enforces the relationships and constraints defined in the first model (see 'XML structure diagram', parses – para 0109, pg. 15 – Note: using a modifiable and dynamic instance of the XML structure diagram to build correspondence between a DTD model and target XML structure reads on using derived markup construct with constraints imposed by markup syntax and pertinent parser -- see Fig. 26B-C; parser– see Fig. 7; Fig. 24; step 2116, Fig. 21B) so to enable implementation of development objects.

As per claim 11, the use of a XML intermediate structure (e.g. para 0109, pg. 1)as a temporary XML model representation to incorporate more intermediate OO data constructs (e.g. see Fig. 25, 26) reads on using a second model to dynamically generate more objects.

As per claims 12-14, refer to the corresponding rejection as set forth in claims 3-5.

As per claims 15-16, Charisius discloses wherein the XML schema includes a tree based on aggregation relationships in the first data model; wherein the XML schema includes a reference based on an association relationship in the first data model (e.g. Fig. 24-25; Figs 26).

As per claim 17, Charisius discloses wherein the XML schema includes a complex type extension based on an inheritance relationship in the first data model (e.g. Fig. 12-18; *JAVA, group... defining elements, "hierarchy"*, para 0124-0126, pg. 18 – Note: UML and Java constructs parsed with construction of AC3 DTD and XML hierarchy reads on inheritance within some complex type in which a group is linked to constituting subelements – see Fig. 25).

Response to Arguments

8. Applicant's arguments filed 6/12/08 have been fully considered but they are not moot in view of the new grounds of rejection.

35 USC § 103(a) Rejection:

(A) Applicants have submitted that 'graphical view of a language representation' as proffered in the Office Action does not teach or suggest 'first model in a first language' (Appl. Rmrks middle, pg. 10). The above allegation does not demonstrate how the claim language of this 'first language' specifically precludes a model as in Charisius from fulfilling (receiving) a model being a particular language; i.e. how this language of Charisius model would be same or not different of any second language. The argument is not persuasive.

(B) Applicants have submitted that Charisius's generating of graphical view TMM where modifications are made to the source code does not teach 'receiving a first model' of claim 1 (Appl. Rmrks pg. 11, top). Charisius modeling is based on fetching file from a UML format,

hence any instance of components in such UML format can be analogized with a model component received into the TMM framework.

(C) Applicants have submitted that 'UML objects' proffered by the Office Action, do not constitute 'constraints for developing the application' (Appl. Rmrks pg. 11, 2nd para). What is interpreted as 'for developing the application' is deemed a broad limitation that includes constraints from the UML constructs and subsequently enlisted into the TMM instance. The above observation thus alleged is not persuasive.

(D) Applicants have submitted that none of data structure, template, or generating of graphical view as proffered in the Office Action teach or suggest 'generating a set of intermediate objects using the model' (Appl. Rmrks pg. 12, 1st , 2nd para). The Office Action has pointed to a variety of structures or OO classes being delivered into the framework as a consequence to the step of creating a instance of TMM based on UML; that is, these structure are considered intermediate set using the TMM instance and the captured UML construct, in order to yield more construct. There are not sufficient details in 'intermediate objects' for the structure(s) by Charisius to be precluded from reading into this limitation. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. The office citing of instance of instance of RWI, IDE or SCI based on Figure 7 of Charisius to illustrate instantiation of each of the component modules of this application program interface, wherein the parser is one component thereof to observe XML construct or checking content being read from the intermediate structure.

(E) Applicants have submitted that none of the portions provided by the Examiner teaches or suggests 'generating an API using the set of intermediate objects … such that the API enforces … enables accessing the development objects' (Appl. Rmrks pg. 12, bottom). The rejection has analogized the API as a composite application interface having several functional modules, and has mentioned Figure 7 as illustrating a basic overview of this API including functional modules components whose instantiation would be instantiated APIs to fulfill this 'deriving' or 'generating' a API' limitation. For example:

First, using the packages from derived template and class symbols represented in UML model, along with graphical view of code objects to generate an metamodel within interface 610, including instance of RWI, IDE or SCI **reads on** API being instantiated using intermediate objects, i.e. an instance of TMM within Interface 610 for further tasks; and

Second, using a modifiable and dynamic instance of the XML structure diagram to build correspondence between a DTD model and target XML structure **reads on** input into a parser component of above API (refer to rejection in claim 1). That is, based on the provided functionality of API Figure 7 and the UML analysis, the APIs instances being derived are either the API calls being invoked for retrieving class or model package components amount to API calls to access components, or API calls being made to read the XML construct, derive a tree or a DTD includng parsing all the pertinent markup hierarchy. The input into the parser API would be analogized to the markup construct, which is an intermediate structure (see Fig. 21B, Fig 24). It is further noted that according to the Specifications (metadata API 130) and the claim language, the API is construed as application interface or a functional metadata component made from derived Java constructs or other intermediate structures, this API to support development of

code, including for example accessing of objects and enforcing constraints. Charisius discloses an API constructed from the elements of the TMM including derived UML constructs, that enables both accessing of OO class objects and enforcing constraints in parsing XML into a deriving a DTD.

The claimed 'generate an API using the set of intermediate objects as inputs … development objects' has been deemed fulfilled; i.e. the argument is not convincing.

(F) Applicants have submitted that Charisius does not teach 'using UML model to … generate a metamodel' (Appl. Rmrks first para, pg. 13). It is not clear whether this observation proves that some claim language has not been met by teachings provided in Charisius. For the sake of argument though, Charisius' disclosing of UML constructs and language (see Fig. 12-19) amounts to more than sufficient teaching as to show that UML constructs are instrumental in providing building blocks and constraints therefor in the Charisius framework using the functional modules of Figure 7 high-level API. The argument fails to prove how specific cited parts of a reference cannot meet a specific language of the claim, hence is largely misdirected.

(G) Applicants have submitted that Charisius does not teach 'generating an API using the set of intermediate objects' (Appl. Rmrks pg 13 middle); this allegation is referred back to the reply set forth in section E above.

(H) Applicants have submitted that a prima facie case of obviousness (Appl. Rmrks pg. 13, bottom) has not been articulated based on the above deficiencies of the Office Action. This is referred back to the above sections.

In all, the claims as amended stand rejected as set forth in the Office Action.

Conclusion

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.
Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

July 06, 2008